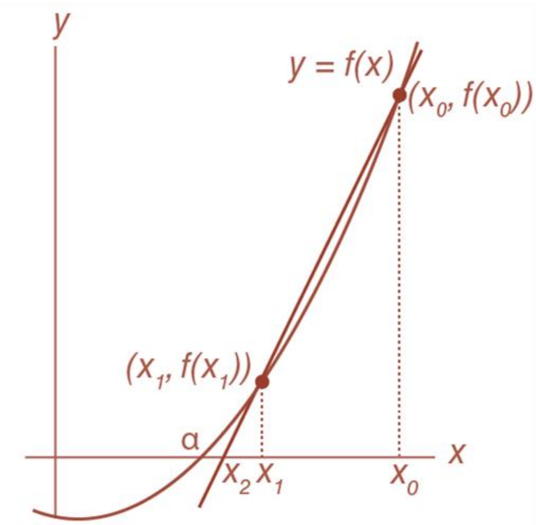
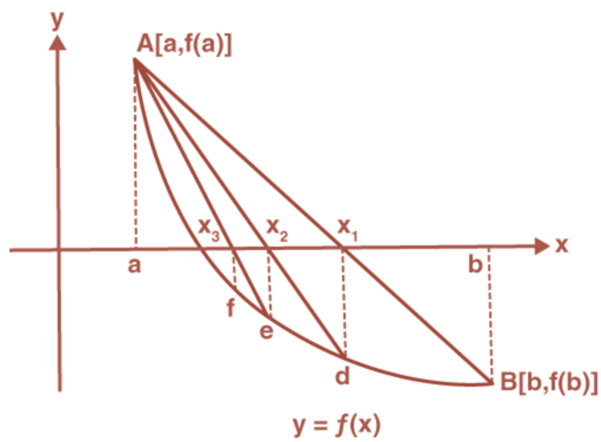
Numerical Analysis

Project 1





Prepared by:

Name	ID	Group
Youssef Samuel Nachaat Labib	6978	2
Youssef Amr Ismail Othman	6913	2
Arsany Mousa Fathy Rezk	6927	2

Pseudocodes for each Method

Bisection Method:

```
xl = -1;
xu = 0;
imax = 500;
Ees = 0.01;

fl = 3*xl^4 + 6.1*xl^3 - 2*xl^2 + 3*xl + 2;
fu = 3*xu^4 + 6.1*xu^3 - 2*xu^2 + 3*xu + 2;

if (fl*fu > 0)
   disp ('does not bracket the root');
   return;
end

for i = 1:imax
   xr = (xl + xu) / 2;
   if (i > 1)
   Ea = abs((xr - xrOld) / xr);
end

  xrOld = xr;
  fl = 3*xl^4 + 6.1*xl^3 - 2*xl^2 + 3*xl + 2;
  fr = 3*xr^4 + 6.1*xr^3 - 2*xr^2 + 3*xr + 2;

  test = fl * fr;
  if (test < 0)
    xu = xr;
  else
    xl = xr;
  end

  if (i > 1)
     if (test == 0)
        Ea = 0;
     end
```

```
      if (Ea < Ees)
         break
     end
   end
end
```

False Position Method:

```
xl = -1;
xu = 0;
imax = 500;
Ees = 0.01;

fl = 3*xl^4 + 6.1*xl^3 - 2*xl^2 + 3*xl + 2;
fu = 3*xu^4 + 6.1*xu^3 - 2*xu^2 + 3*xu + 2;

if (fl*fu > 0)
   disp ('does not bracket the root');
   return;
end

for i = 1:imax
   fl = 3*xl^4 + 6.1*xl^3 - 2*xl^2 + 3*xl + 2;
   fu = 3*xu^4 + 6.1*xu^3 - 2*xu^2 + 3*xu + 2;
   xr = ((xl * fu) - (xu * fl)) / (fu - fl);

   if (i > 1)
      Ea = abs((xr - xrOld) / xr);
   end

   xrOld = xr;

   fr = 3*xr^4 + 6.1*xr^3 - 2*xr^2 + 3*xr + 2;

   if (fr < 0)
      xl = xr;
   else
      xu = xr;
   end
```

```
    if (i > 1)
       if (fr == 0)
          Ea = 0;
       end

       if (Ea < Ees)
          break
       end
    end
end
```

Fixed Point Method:

```
xold = 0.5;
imax = 500;
Ees = 0.01;
syms x
g =  3 / (x-2);
for i = 1:imax
    xnew = subs(g, xold);

    if (i > 1)
       Ea = abs((xnew - xold) / xnew);
    end

    xold = xnew;
    test = subs(g, xnew);

    if (i > 1)
       if (test == 0)
          Ea = 0;
       end

       if (Ea < Ees)
          break
       end
    end
end
```

Newton Raphson's Method:

```
xold = -1;
imax = 500;
Ees = 0.01;

syms x
s =  'x.^5 + 7*x.^4 - 2*x.^2 + sin(x*(pi/pi))';
s_fun = str2func(['@(x)' s])                    % Not Vectorized (Illustration
Only)
s_funv = str2func(['@(x)' vectorize(s)])              % Vectorized
x = linspace(-10, 10, 25);
figure(1)
plot(x, s_funv(x), '-p')
grid
A=str2sym(c);
subs(A,1)
B =  diff (A);
subs(B, 1);

for i = 1:imax
   xnew = xold - (subs(A, xold) / subs(B, xold));

   if (i > 1)
      Ea = abs((xnew - xold) / xnew);
   end

   xold = xnew;
   test = subs(A, xnew);

   if (i > 1)
      if (test == 0)
         Ea = 0;
      end

      if (Ea < Ees)
         break
      end
   end
end
```

Secant Method:

```
xi=-1;
xold=-5;
imax=100;
es=10^(-2);
 syms x;
 A =  x.^5 + 7*x.^4 - 2*x.^2;

 for i=1:1:imax

    fxi=subs(A, xi);
    fxold=subs(A, xold);
    xnew=xi-(fxi*(xold-xi))/(fxold-fxi);
    ea = abs((xnew-xi)/xnew);
    xold=xi;
    xi=xnew;
    test=subs(xnew,A);

    if(test==0)
       ea=0;
       break;
    end

    if (ea < es)
       break;
    end

 end
```

Data Structures Used and Its Implementation Benefits

- Mainly we used arrays only in our implementations to collect the information needed to draw our table that contains the iterations and the accompanied values for each iterations.
- This array mentioned is called “vector” in our code implementation.

Analysis for Each Method Behaviour

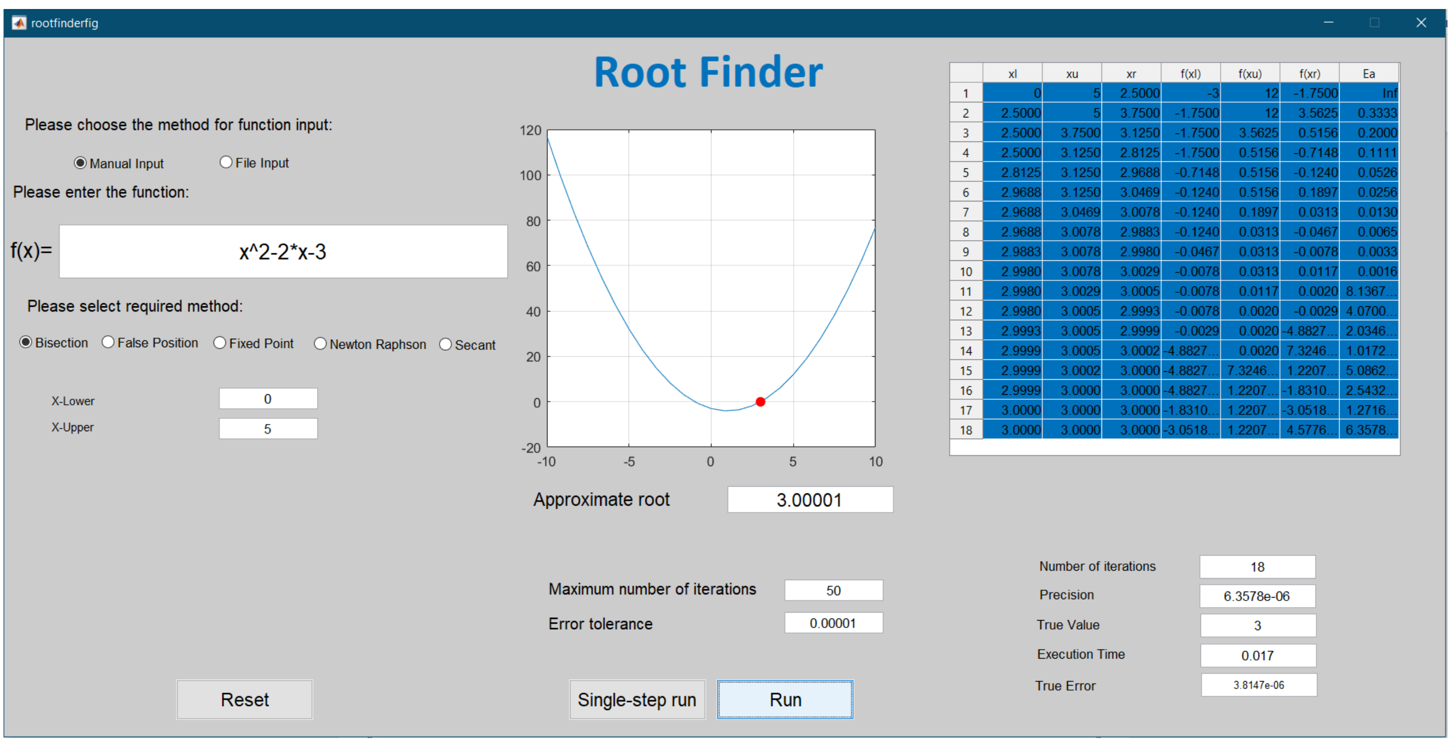
Example1:

$f(x) = x^2 - 2x - 3$

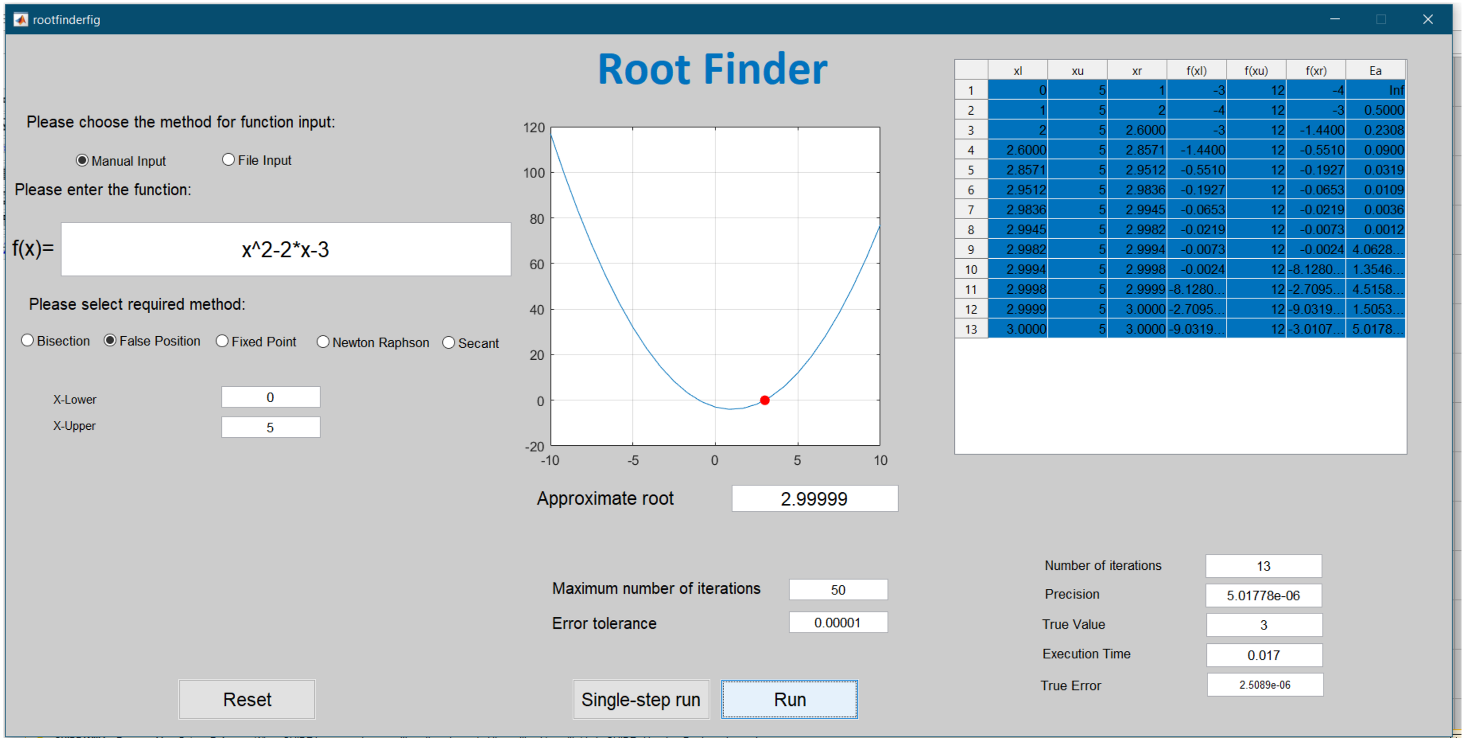
This function has 2 true roots : ($x = 3$ and $x = -1$)

Bracketing methods:

Bisection method:

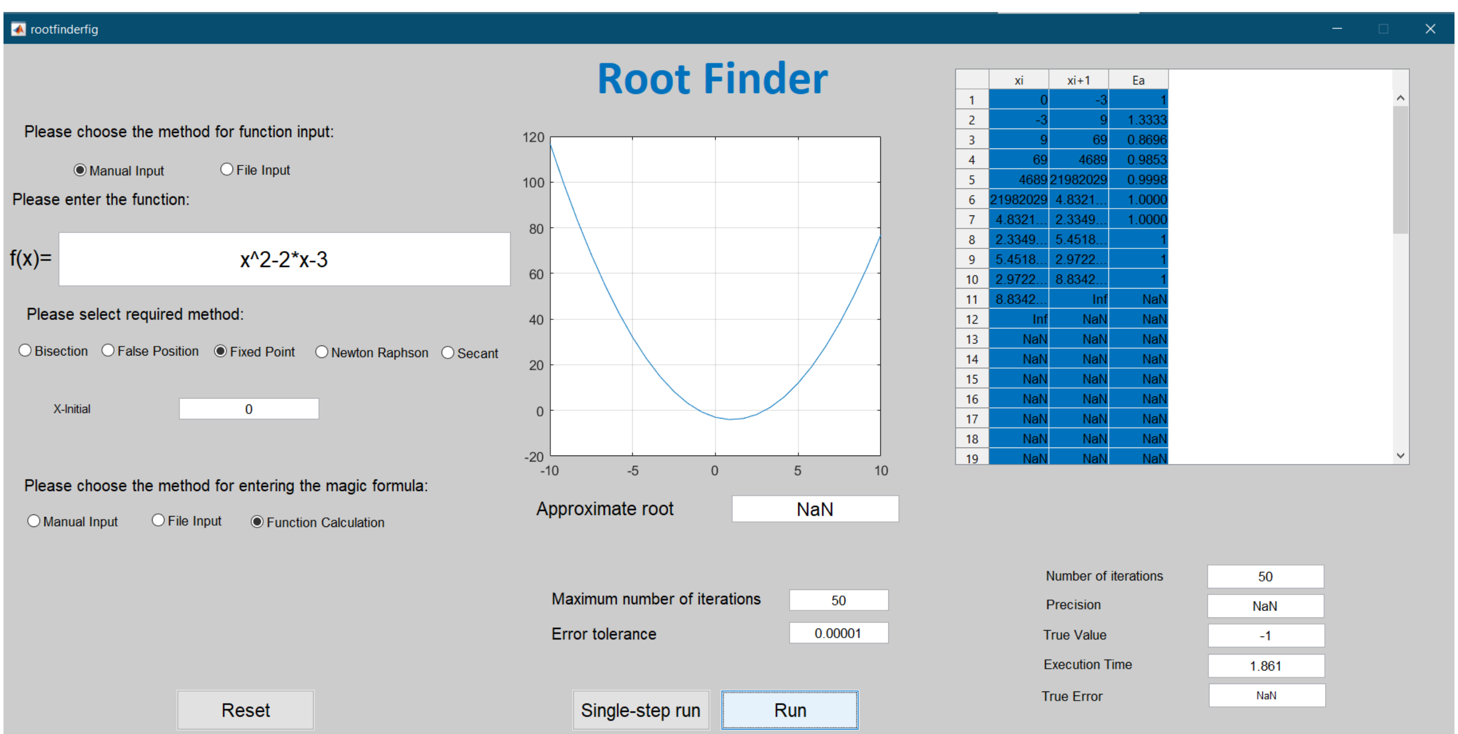


False Positiion method:

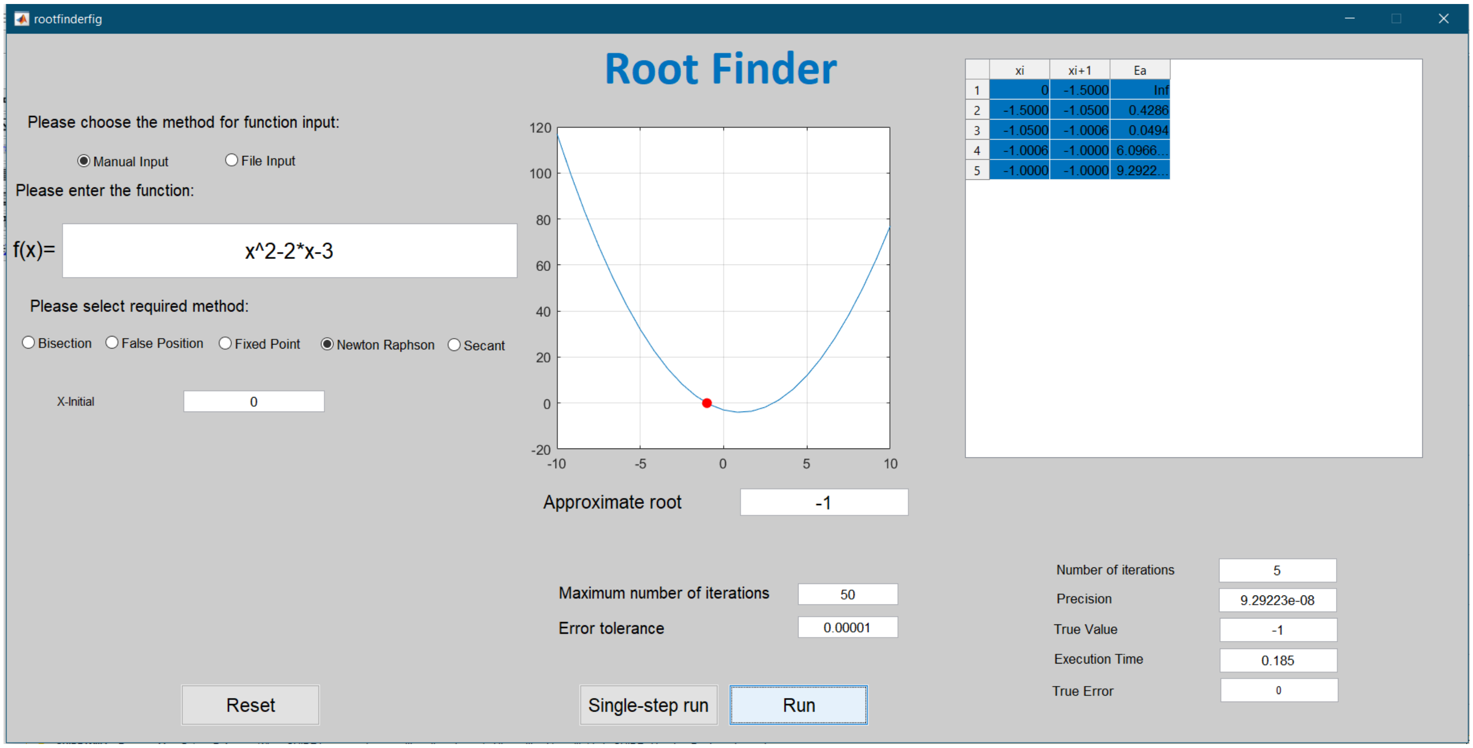


Open methods:

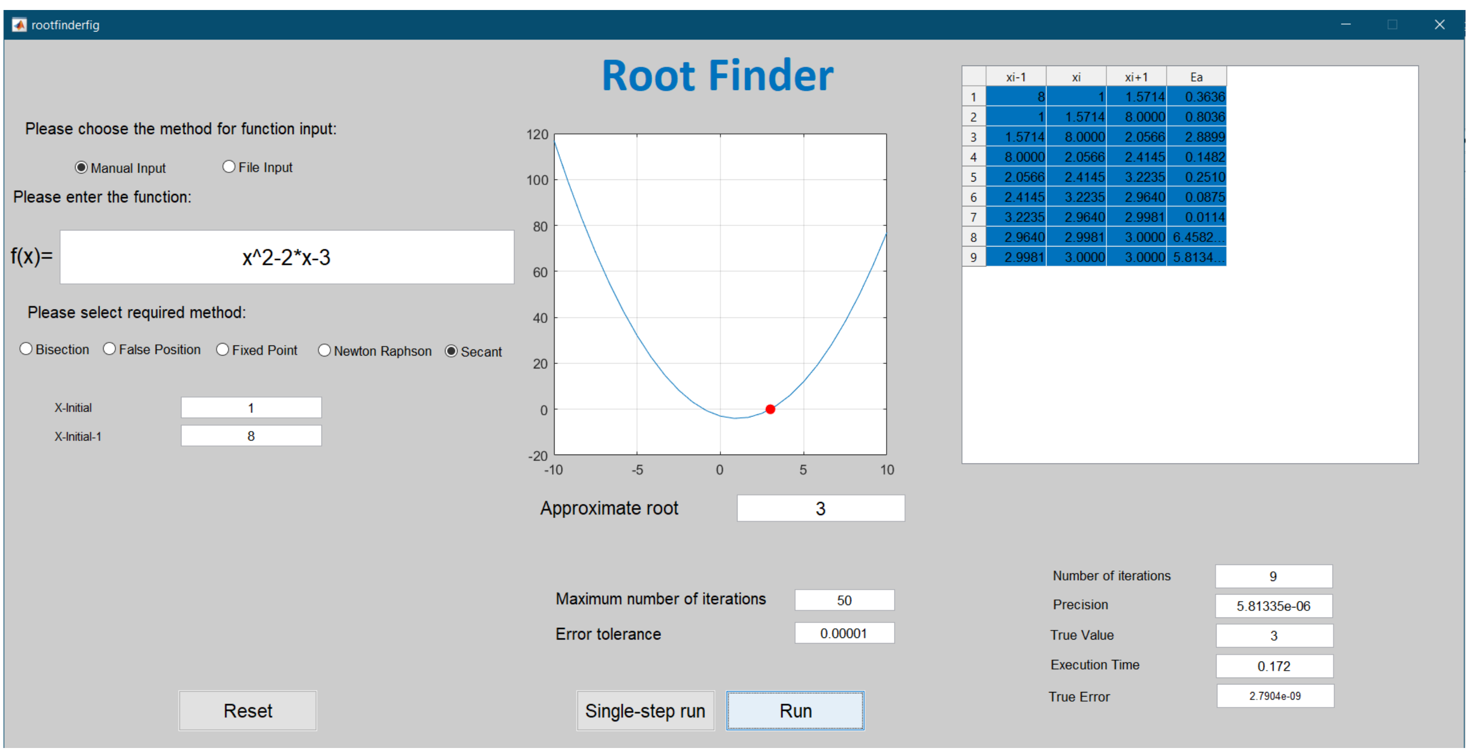
Fixed Point:



Newton Raphson:



Secant:



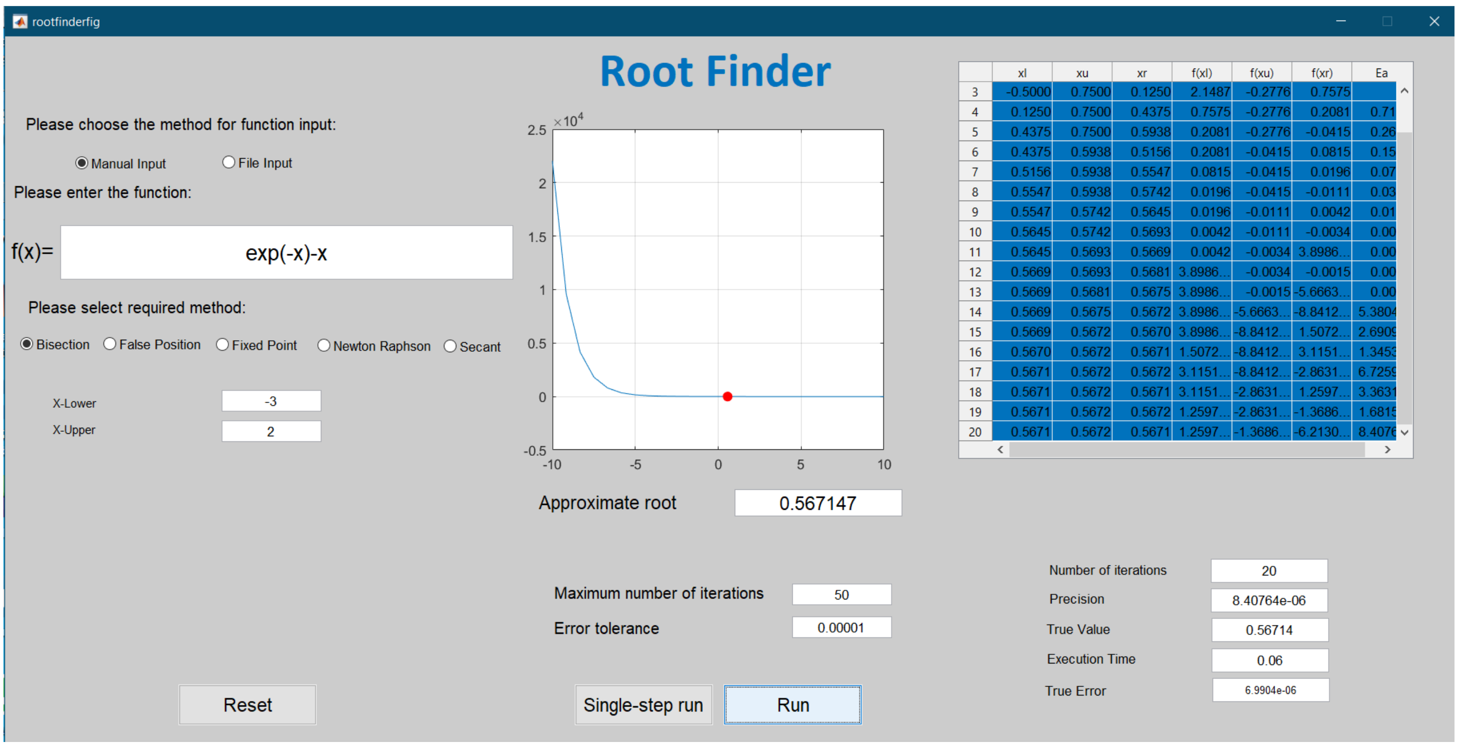
Example 2:

$f(x) = e^{-x} - x$

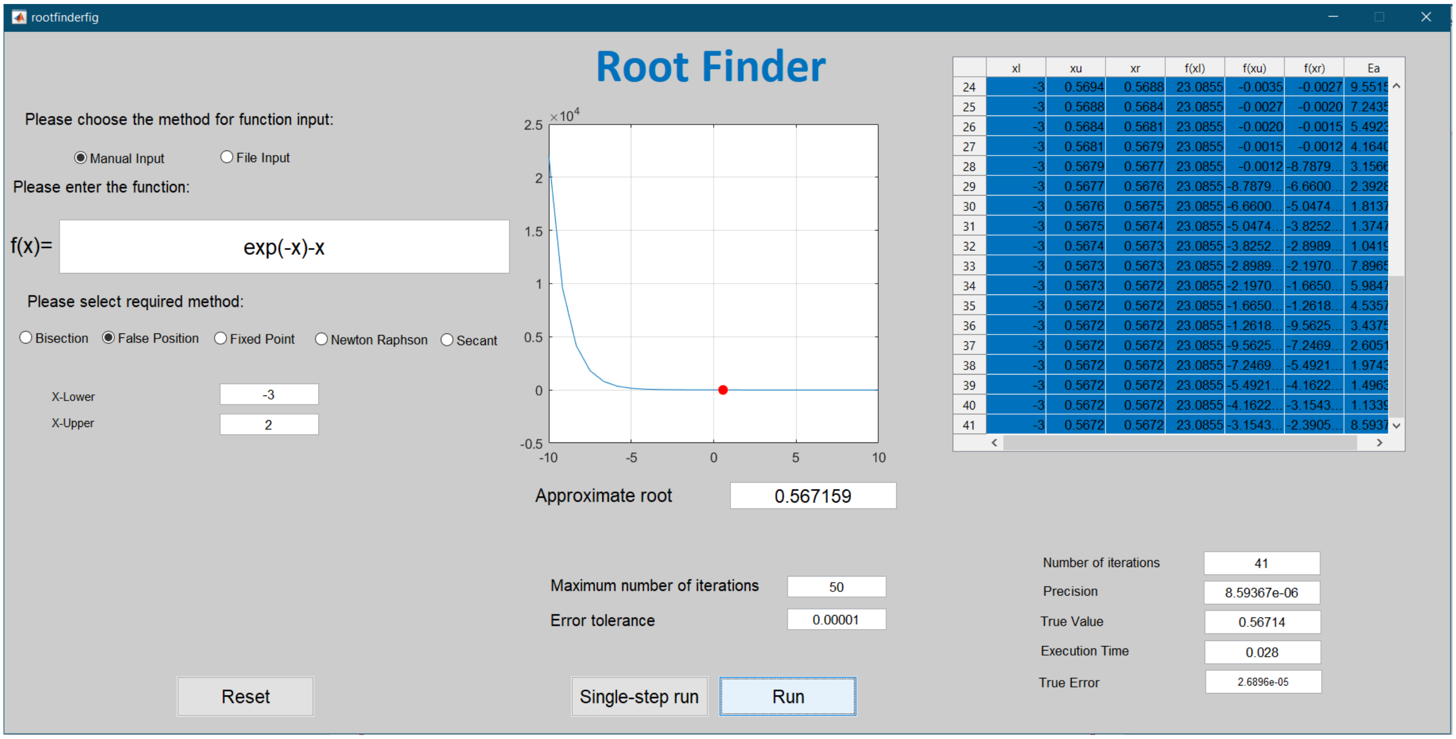
This function has only 1 true roots : (x = 0.56714329)

Bracketing methods:

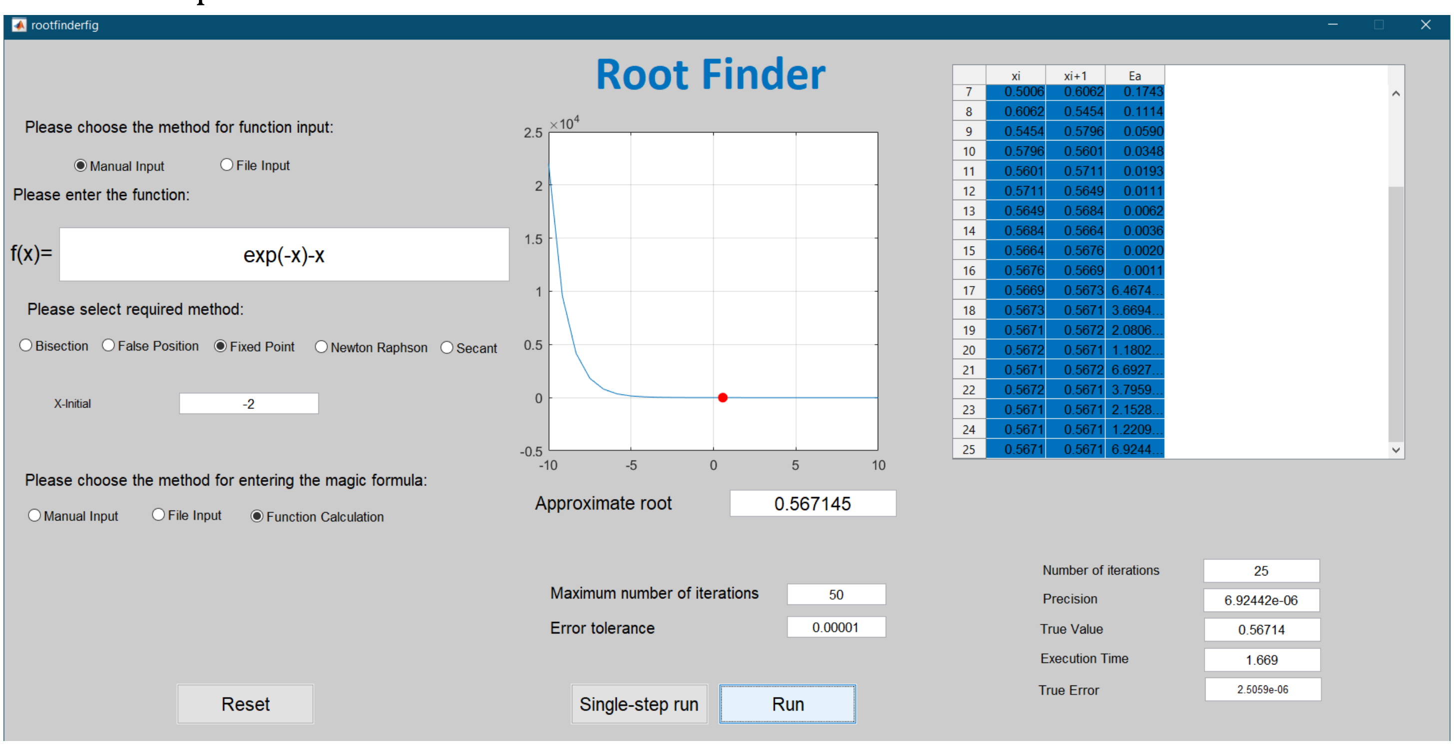
Bisection method:



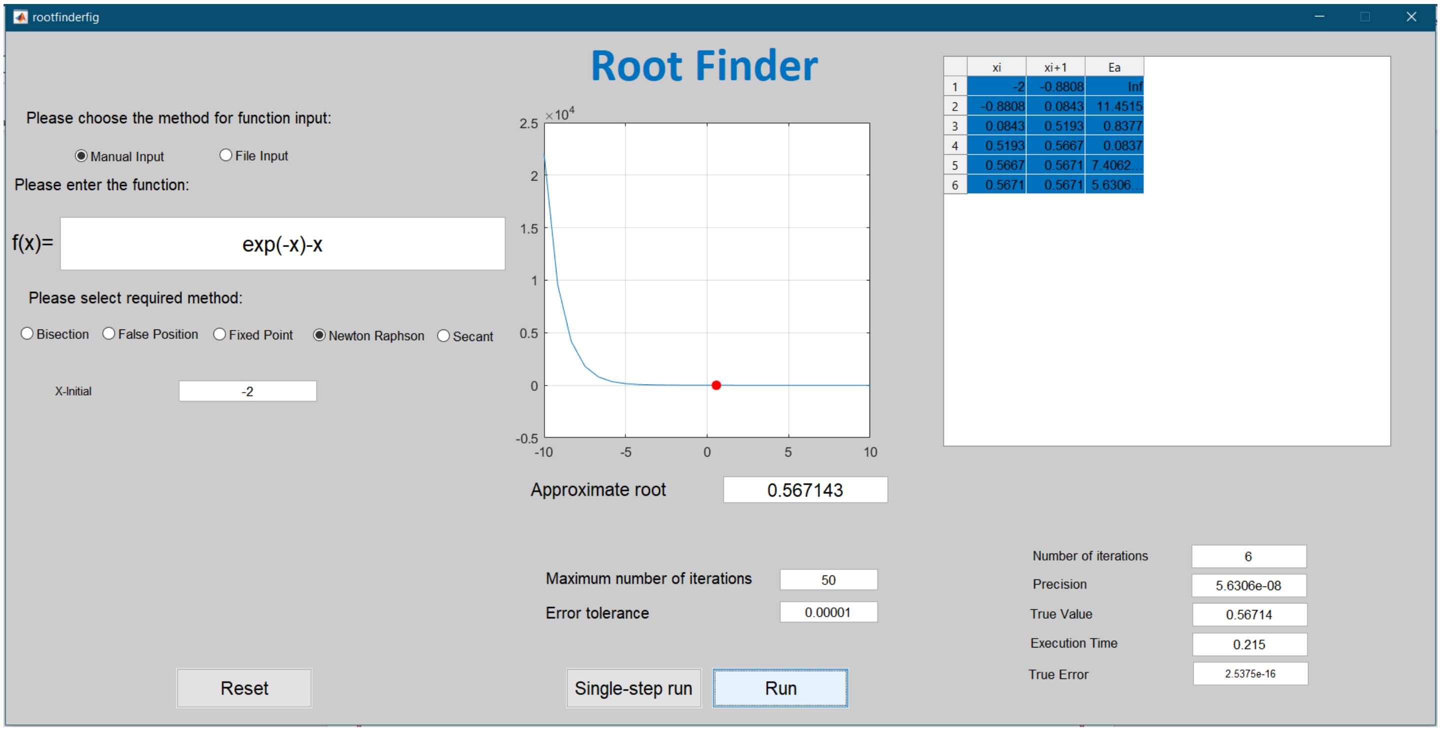
False Position:



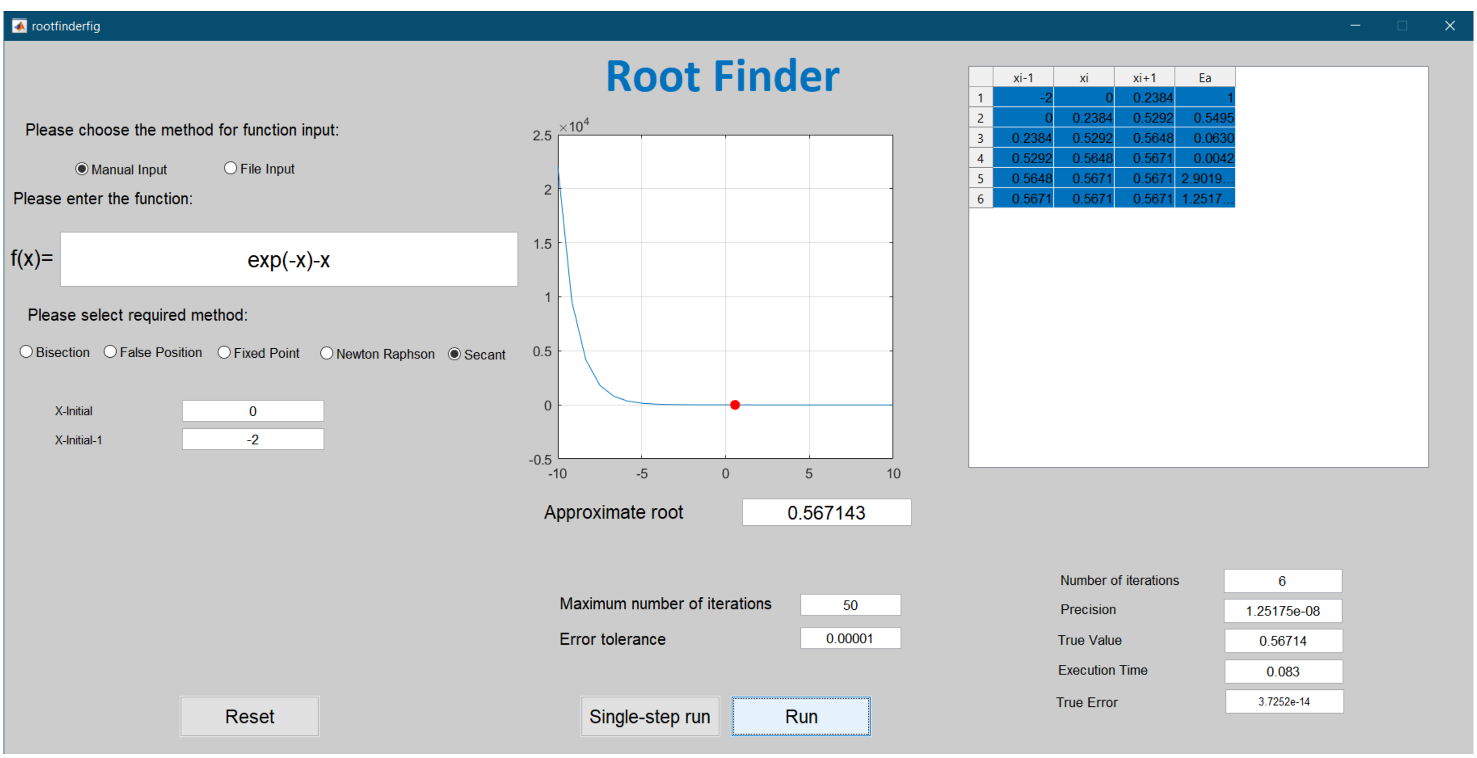
Fixed point:



Newton Raphson:



Secant:



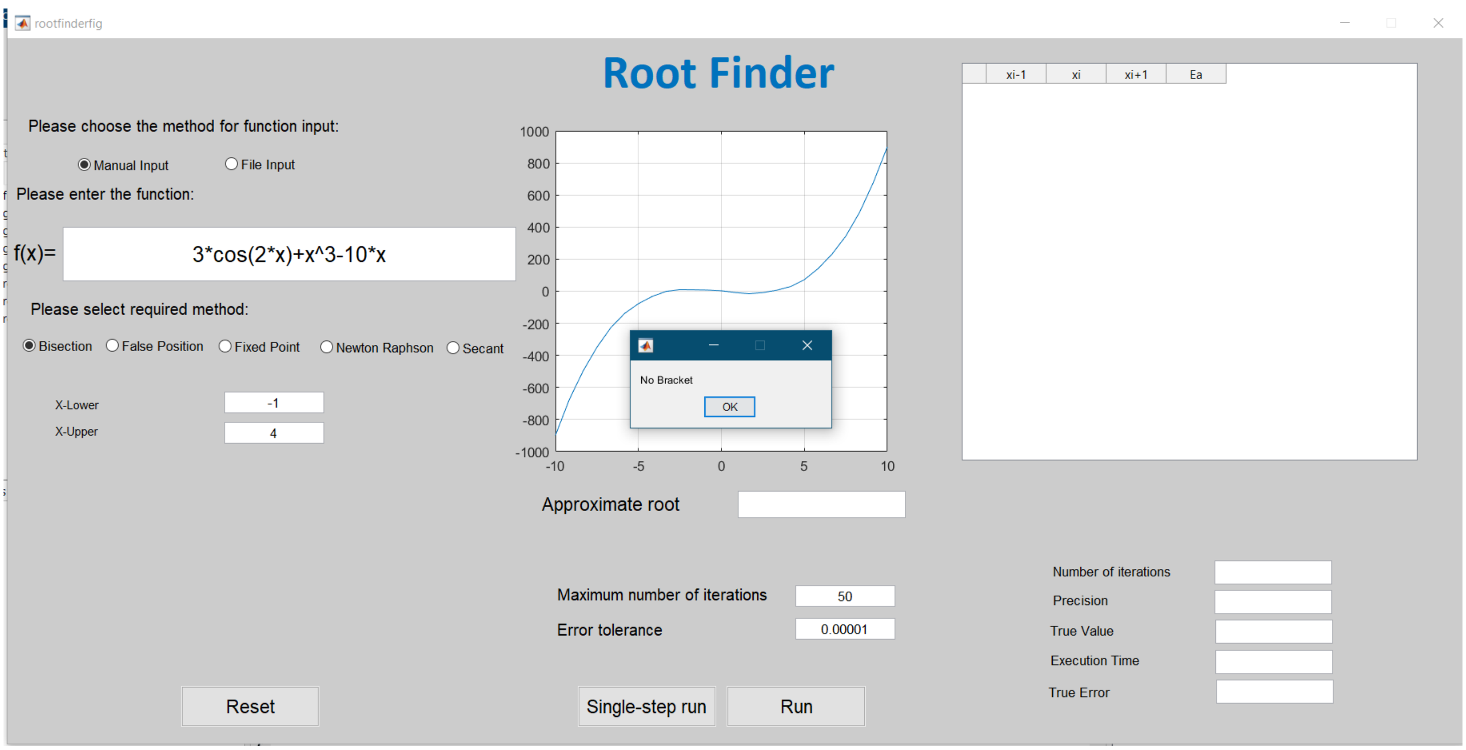
Example 3:

$f(x) = 3\cos(2x) + x^3 - 10x$

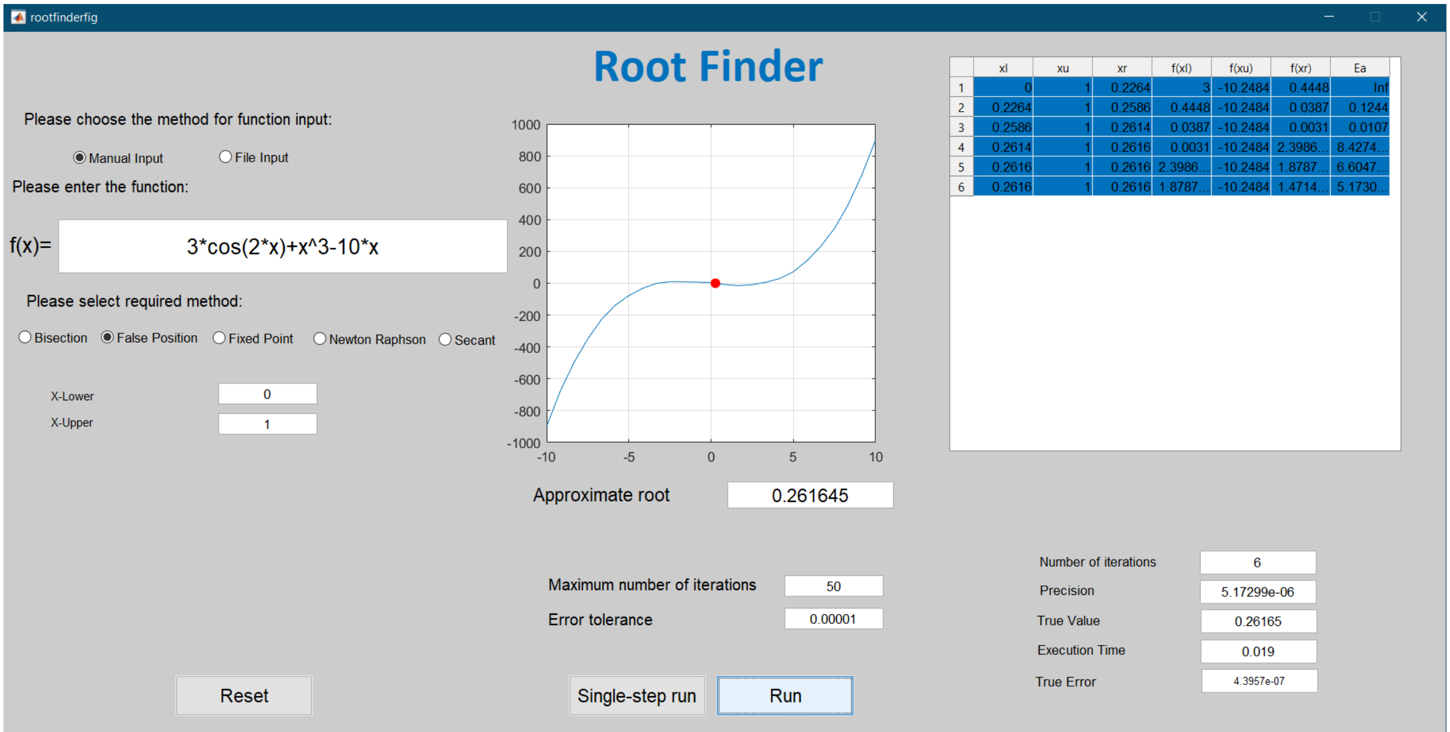
This function has only 1 true roots : (x = 0.3027584109)

Bracketing methods:

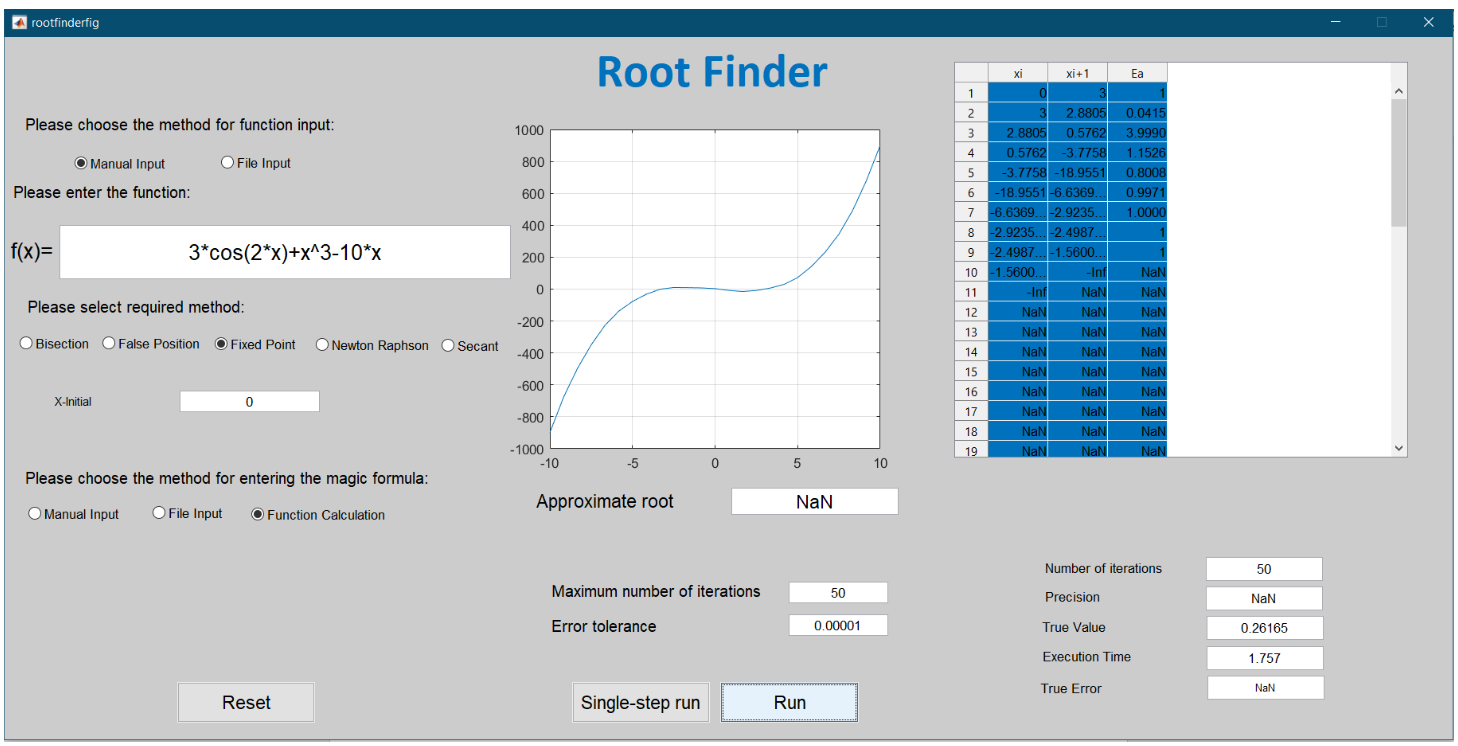
Bisection method:



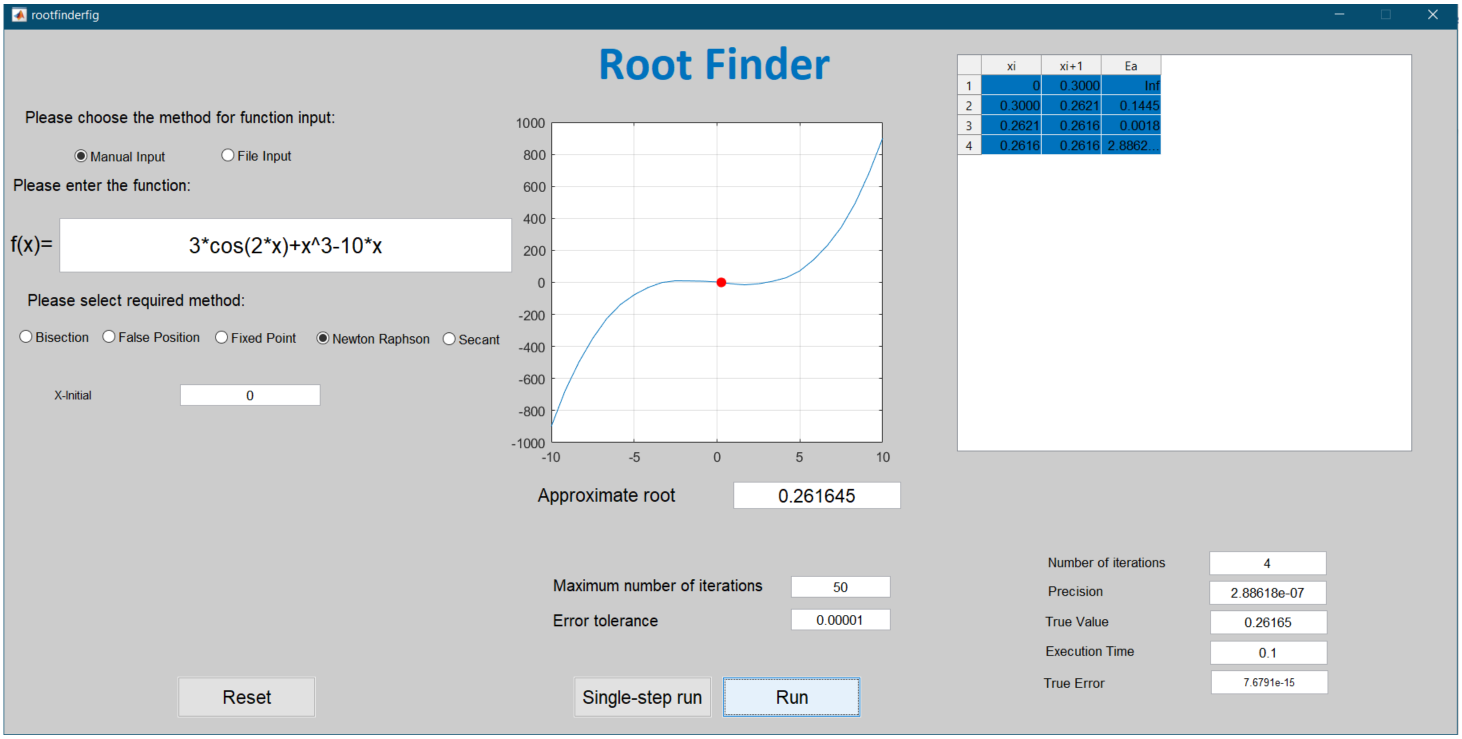
False position:



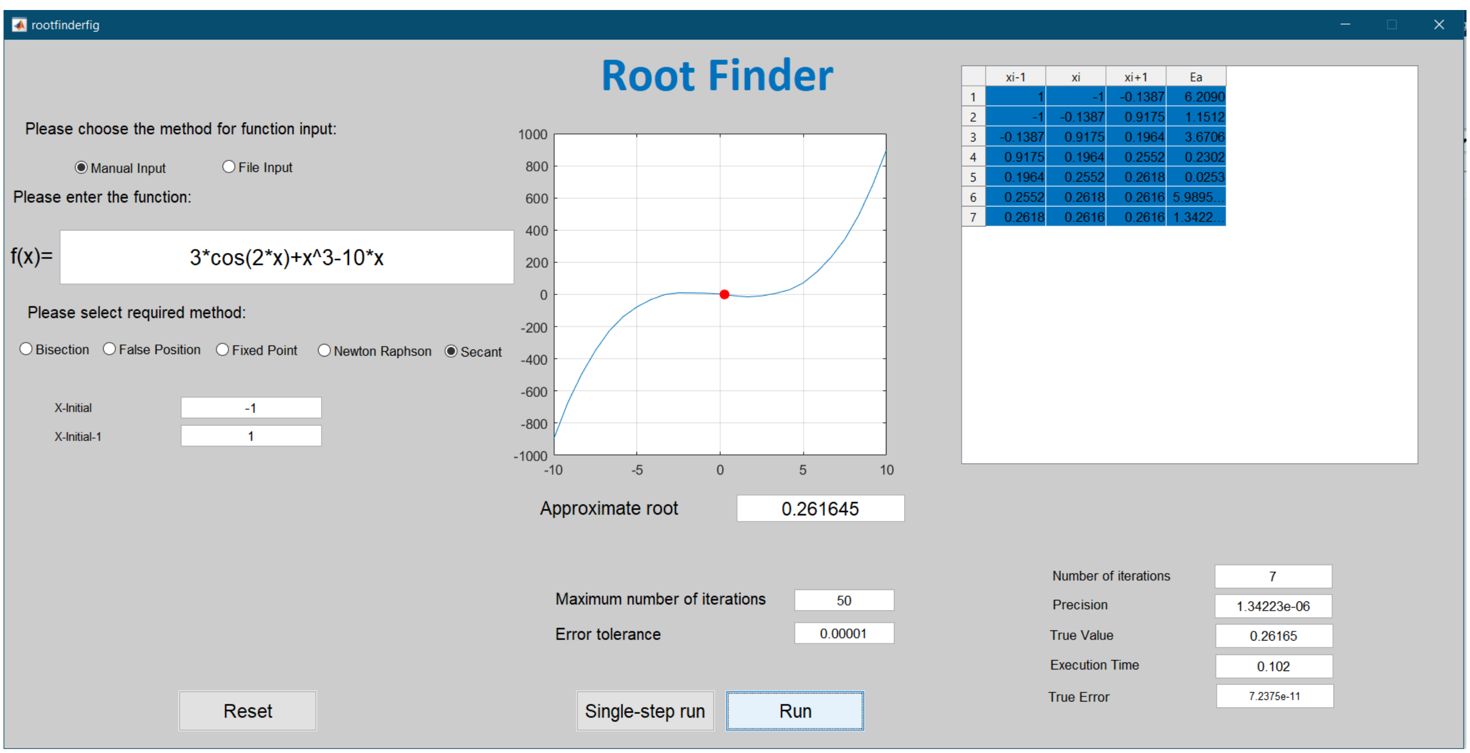
Fixed point:



Newton Raphson:



Secant:



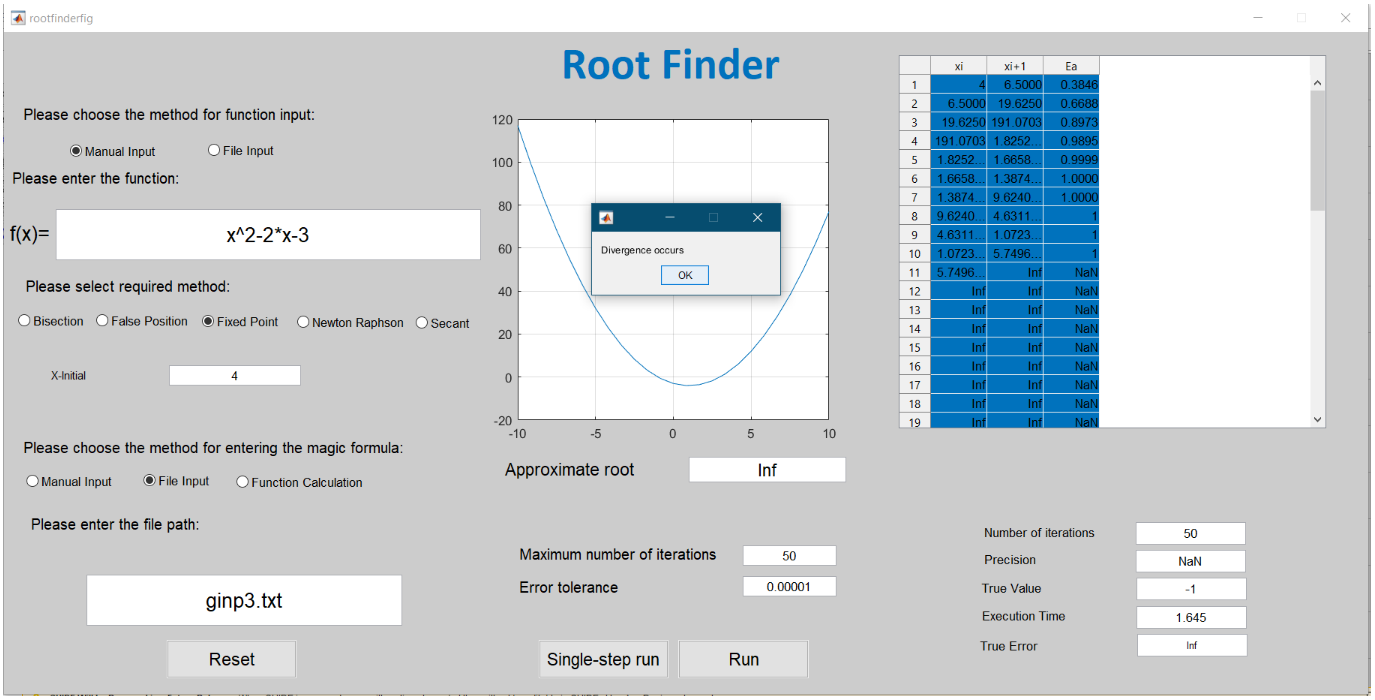
Conclusions:

1. Bracketing methods always converge, unlike the open methods.
2. When open methods converge, it converges faster than the bracketing methods.
3. Open methods can locate multiple roots, while bracketing cannot.
4. Both Bisection and False Position needs 2 initial guesses.
5. Bisection method is generally slower than false position.
6. In functions with one sided nature as in example 2 bisection method is faster than false position.
7. If initial guesses are not appropriate enough it might not find the roots since the check results in no bracketing.
8. Fixed point method need an appropriate magic function g(x) to converge and it converges linearly.
9. Newton Raphson's method is very efficient and has a quadratic order of convergence.
10. In Newton Raphson's method, if the function has zero slope it causes division by zero (not shown in the previous 3 examples).
11. Secant method is very similar to Newton Raphson's method but it can deal easily with functions that have difficulties in its differentiation.

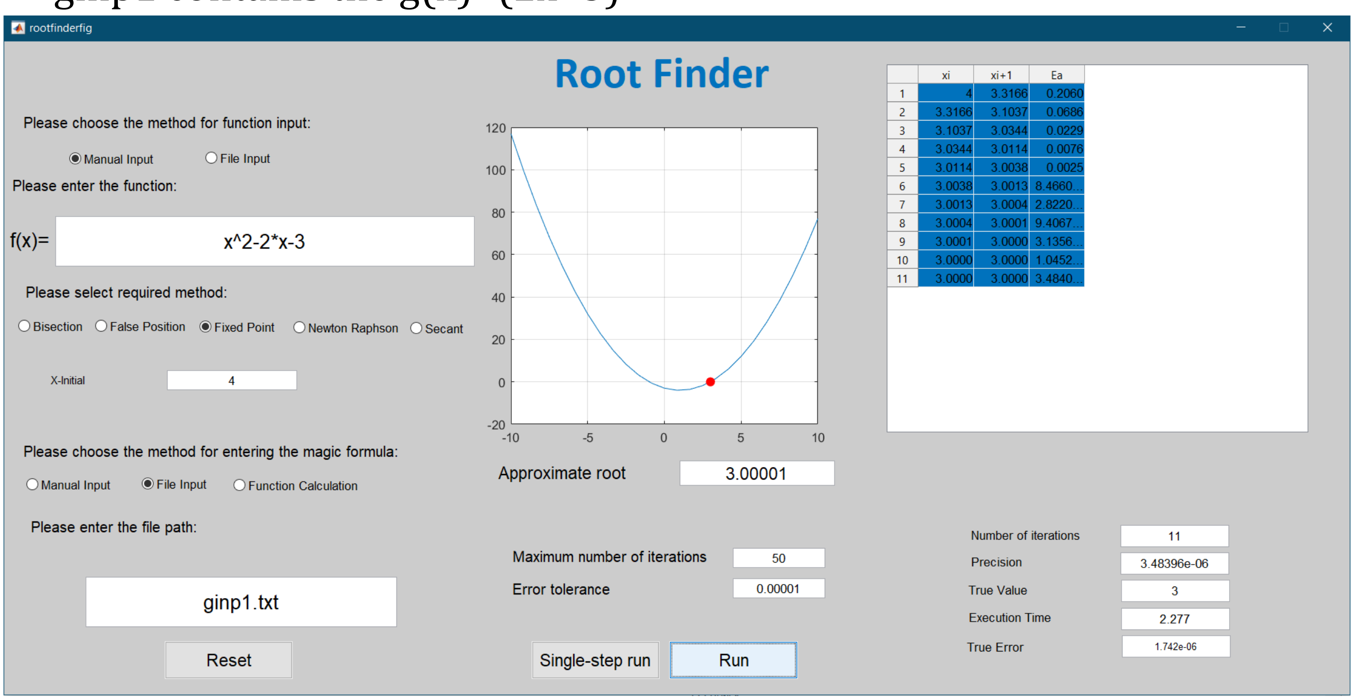
Problematic Functions and Reasons of this Misbehavior

1- In fixed point iterations the choice of the magical formula should be done wisely to avoid divergence.

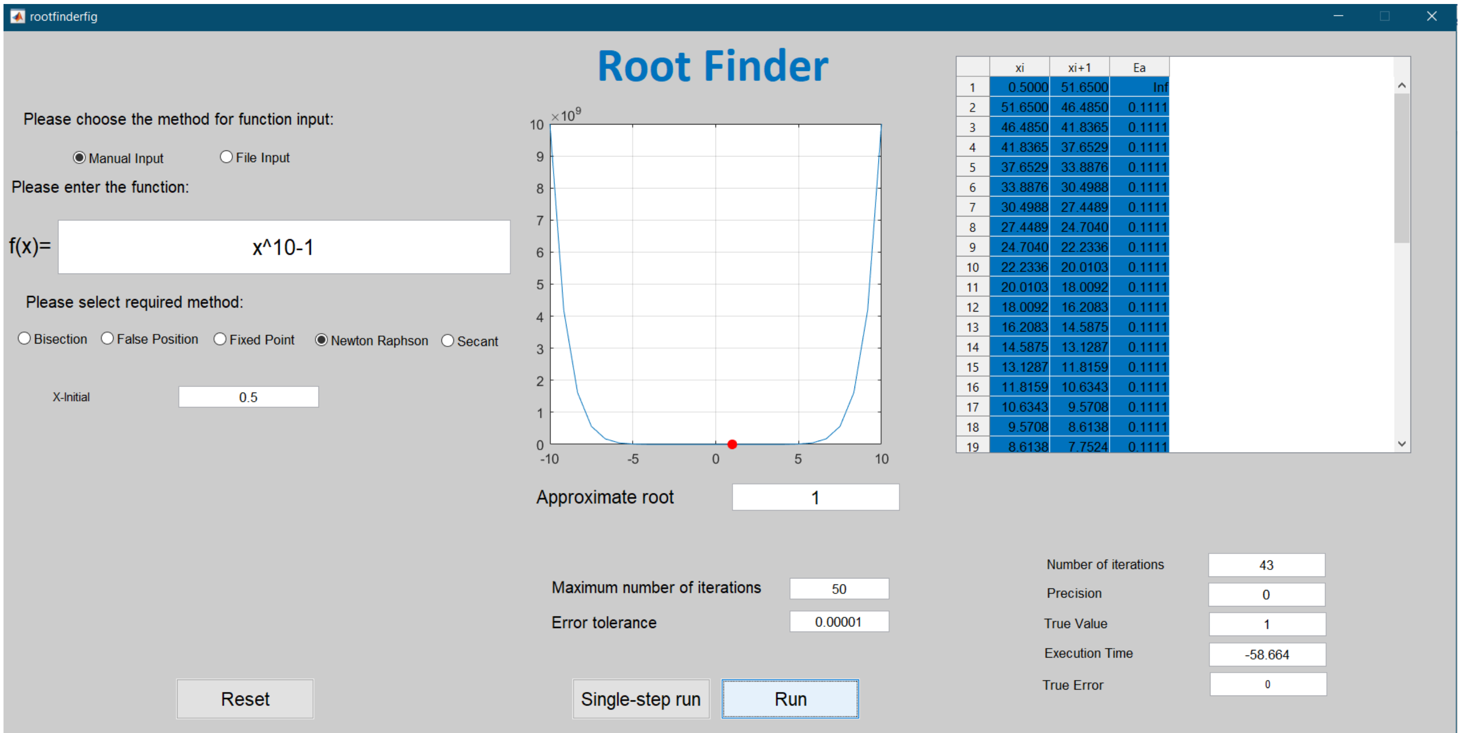
ginp3 contains the $g(x)=(x^2-3)/2$



ginp1 contains the $g(x)=(2x+3)^{0.5}$



2- Newton Raphson's method might be too slow like this case



3- Also in Newton Raphson's method division by zero my occur due to zero slope.(This could be handled using secant method)